PATENTTI- JA REKISTERIHALLITUS
NATIONAL BOARD OF PATENTS AND REGISTRATION

Helsinki 29.4.2003

E T U O I K E U S T O D I S T U S
P R I O R I T Y D O C U M E N T

Hakija Applicant	**Nokia Oyj** **Helsinki**
Patenttihakemus nro Patent application no	**20020414**
Tekemispäivä Filing date	**04.03.2002**
Kansainvälinen luokka International class	**G06N**

Keksinnön nimitys
Title of invention

"Mechanism for unsupervised clustering"
(Mekanismi valvomatonta klusterointia varten)

Täten todistetaan, että oheiset asiakirjat ovat tarkkoja jäljennöksiä Patentti- ja rekisterihallitukselle alkuaan annetuista selityksestä, patenttivaatimuksista, tiivistelmästä ja piirustuksista.

This is to certify that the annexed documents are true copies of the description, claims, abstract and drawings originally filed with the Finnish Patent Office.

Pirjo Kaila
Tutkimussihteeri

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN COMPLIANCE WITH RULE 17.1(a) OR (b)

Maksu 50 €
Fee 50 EUR

Maksu perustuu kauppa- ja teollisuusministeriön antamaan asetukseen 1027/2001 Patentti- ja rekisterihallituksen maksullisista suoritteista muutoksineen.

The fee is based on the Decree with amendments of the Ministry of Trade and Industry No. 1027/2001 concerning the chargeable services of the National Board of Patents and Registration of Finland.

Osoite: Arkadiankatu 6 A
P.O.Box 1160
FIN-00101 Helsinki, FINLAND
Puhelin: 09 6939 500
Telephone: + 358 9 6939 500
Telefax: 09 6939 5328
Telefax: + 358 9 6939 5328

MECHANISM FOR UNSUPERVISED CLUSTERING

BACKGROUND OF THE INVENTION

The invention relates to clustering techniques that are generally used to classify input data into groups or clusters without prior knowledge of those clusters. More particularly, the invention relates to methods and apparatus for automatically determining the cluster centres. An example of such clustering techniques is a Self-Organizing Map, originally invented by Teuvo Kohonen. The SOM concept is well documented, and a representative example of a SOM application is disclosed in US patent 6 260 036.

The current framework under investigation for describing and analyzing context has a critical component based on clustering of data. This clustering is expected to appear at every stage of context computation, from the processing of raw input signals to the determination of higher order context. Clustering has been well studied over many years and many different approaches to the problem exist. One of the main problems is knowing how many clusters exist in the data. Techniques exist to estimate the number of clusters in a data set, however the methods either require some form of a priori information or assumptions on the data, or they estimate the number of clusters on the basis of an analysis of the data, which may require storing the data, as well as being computationally demanding. None of these approaches seem entirely suitable for on-line, unsupervised cluster analysis in a system with limited resources as would be the case for a context aware mobile terminal. With this in mind, and in what follows an algorithm for on-line cluster analysis which is computationally simple and needs no assumption about the number of clusters in the data being analyzed, is described.

Clustering is an important part of any data analysis or information processing problem. The idea is to divide a data set into meaningful subsets so that points in any subset are closely related to each other and not related to points in other subsets. The definition of related may be as simple as the distance between points. Many different approaches and techniques can be applied to achieve this goal. Each approach has its own assumptions and advantages and disadvantages. One of the best-known methods from the partition-based clustering class is the K-means algorithm, which adaptively tries to position K 'centres' which minimize the distance between the input data vectors and the centres. One of its disadvantages is that the number of the K

centres must be specified before the clustering is attempted. In the case of an unknown data set this may not always be possible. The algorithm can be run several times with different values of K and the optimum K is chosen on the basis of some criteria. For an on-line system where the data is not stored, this approach is slow and impractical.

Thus a problem associated with the known clustering techniques is that while it is relatively easy for humans to determine the cluster centres, such a determination is difficult for computers.

BRIEF DESCRIPTION OF THE INVENTION

An object of the invention is to provide a method and an apparatus for implementing the method so as to alleviate the above disadvantages. In other words, the object of the invention is to provide a method for automatically determining the cluster centres, such that the method is easily implemented in a computer system.

The object of the invention is achieved by a method and an arrangement which are characterized by what is stated in the independent claims. The preferred embodiments of the invention are disclosed in the dependent claims.

A computer-implemented method according to the invention can be implemented by the following steps:

initializing a first data structure that comprises a lattice structure of weight vectors that create an approximate representation of a plurality of input data points;

performing a first iterative process for iteratively updating the weight vectors such that they move toward cluster centres;

performing a second iterative process for iteratively updating a second data structure utilizing results of the iterative updating of the first data structure; and

determining the weight vectors that correspond to cluster centres of the input data points on the basis of the second data structure.

A preferred embodiment of the invention is based on the following idea. Self-organizing maps generally use a lattice structure of nodes and associated with each node is a weight vector. Each data point in the input data is iteratively compared with each weight vector of the lattice, and the node whose weight vector best approximates the data point is chosen as the winner

for that data point and iteration. Then the weight vectors associated with each node of the lattice are adjusted. The adjustment affected to each node's weight vector is dependent on the winning node through a neighbourhood function. Following the adjustment of the weight vectors a next iteration step is taken.

As used in this context, the term 'neighbourhood function' is a function of distance on the lattice between the winning node and the node being updated such that the value of the function generally decreases as the distance increases. With normalized SOMs, the value of the function is unity for a distance of zero. A common form for the neighbourhood function is Gaussian, but preferred embodiments of the invention make use of neighbourhood functions that are not strictly Gaussian.

In addition to the primary iteration process for updating the SOM or other clustering mechanism, a second iterative process is run, and the second iterative process gives a numerical value for the lattice nodes such that the numerical value increases if the node's weight vector is positioned at a cluster centre. Then the cluster centres are determined, not on the basis of the weight vectors, but on the basis of the numerical values produced by the second iterative process.

Thus the problem of locating cluster centres reduces to a relatively straightforward problem of locating local maxima in the numerical values produced by the second iterative process.

An advantage of the invention is that it is much easier for machines to locate local maxima in the numerical values than to locate cluster centres in the clustering mechanism wherein the cluster centres are the location in which the density of the weight vectors is highest.

In a preferred embodiment of the invention, the second data structure comprises a coefficient for each of the weight vectors in the lattice structure. Each iteration in the first iterative process comprises selecting a winner weight vector for each of the data points on the basis of a distance measure between the input data point and the weight vector. Each iteration in the second iterative process comprises calculating a next value of each coefficient on the basis of the current value of the coefficient; and a combination of: 1) the current coefficient of the winner weight vector, 2) a second neighbourhood function that approaches zero as the distance on the lattice structure between the weight vector and the winner weight vector increases, and 3) an adjustment factor for adjusting convergence speed

between iterations.

The combination referred to above can be a simple multiplication.

If the second neighbourhood function is selected appropriately, such that the second data structure has distinct borders, the step of determining the weight vectors can be accomplished simply by selecting local maxima in the second data structure.

A preferred version of the second neighbourhood function is not monotonous, giving negative values at some distances. Also, the second neighbourhood function is preferably made more pronounced over time, as the number of prior iterations increases.

Preferably, the first data structure is or comprises a self-organizing map and the input data points represent real-world quantities.

BRIEF DESCRIPTION OF THE DRAWINGS

In the following the invention will be described in greater detail by means of preferred embodiments with reference to the attached drawings, in which

Figure 1 illustrates a self-organizing map (SOM) with six clusters of input data points;

Figure 2 shows a typical form of the neighbourhood function used in a SOM algorithm;

Figure 3 shows a 15 by 15 lattice structure resulting from uniformly-distributed input data;

Figure 4 shows a probabilistic map for visualizing the cluster centres in a SOM with six cluster centres;

Figure 5 shows a preferred form of the second neighbourhood function used in the second iterative process according to a preferred embodiment of the invention;

Figure 6 shows a computer pseudocode listing for generating the function shown in Figure 5.

Figure 7 shows a coefficient map that visualizes the data structure used for locating the cluster centres in the SOM; and

Figure 8 is a flow chart illustrating a method according to the invention wherein the method comprises two iterative processes run in tandem;

Figures 9 and 10 show another pair of SOM map and coefficient

map, respectively, with five clusters;

Figures 11 and 12 show a SOM and coefficient map, respectively, for an exceptional distribution of input data;

Figure 13 shows an example of a neighbourhood function used in an automatic cluster-labelling algorithm;

Figure 14 shows the result of the automatic cluster-labelling algorithm; and

Figure 15 shows how the automatic cluster-labelling algorithm can be integrated with the cluster-determination algorithm according to the invention.

DETAILED DESCRIPTION OF THE INVENTION

A practical example of the invention is disclosed in the context of self-organizing maps. A SOM is a learning algorithm or mechanism from the area of Artificial Neural Networks (ANNs) that find wide application in the area of vector quantization, unsupervised clustering and supervised classification. Reasons for its widespread use include its robustness, even for data sets of very different and even unknown origin, as well as the simplicity of its implementation. The SOM uses a set of weight vectors to form a quantized, topology-preserving mapping, of the input space. The distribution of the weights reflects the probability distribution of the input. The SOM representation is used in clustering applications generally by clustering the weight vectors after training, using for example the K-means algorithm. However the problem of the original K-means algorithm still remains, that is, determining the value of K the number of centres. In the following a method, based on the SOM algorithm, is described which can be used to automatically determine cluster centres in an unsupervised manner. That is the number of clusters does not have to be predefined and groups of adjacent SOM weight vectors represent the cluster centres. Unlike the K-means algorithm where each cluster is represented by one centre, in the inventive algorithm the cluster is represented by a set of centres which correspond to weight vectors in the SOM. The algorithm requires few additional computational resources and makes direct use of the information generated during the learning of the SOM. It is already clear how the algorithm can be considered a hybrid of the K-means algorithm and a probabilistic mixture model based method. Each cluster is represented by a set of centres, which correspond to a set of weights in the

SOM. In turn the SOM weights form an approximation of the probability distribution of the input.

From the ANN point of view it may be interesting as the algorithm uses lateral inhibition between the weight vectors to generate the clusters and a form of Hebbian learning. It is clear that the performance of the clustering is heavily dependent on the topology preserving and convergence ability of the SOM.

The SOM algorithm

Figure 1 shows a self-organizing map 10. More particularly, Figure 1 shows an online version of SOM. There also exists a "Batch SOM" but this requires storing all the input points and going through all of them several times which is why the online-version is preferred here. Reference numerals 11 generally denote input data points that in most SOM applications represent real-world events or quantities. The SOM algorithm creates a SOM or lattice structure 12 by means of an iterative process that can be summarized as follows. Consider a time sequence of inputs $\omega(t)$, $t = 1, \ldots$, with $\omega \in R^m$ and a probability distribution p_ω. The SOM itself consists of a total of N weight vectors $X \in R^m$ distributed on an n-dimensional lattice. Thus there are two associated dimensions, the dimension m of the input data and the weight vectors, and the dimension n of the lattice. The reason for having a lattice is to be able to define neighbourhood relations between adjacent weights. For example, if each weight k has an associated position vector $i_k \in I^n$ on the lattice, then the distance $d_L(i_j, i_k)$ between weights k and j on the lattice can be defined. The initial values of the weight vectors can be randomly chosen, as the convergence of the algorithm is independent of the initial conditions. At each iteration, the distance $d(\omega(t), X_k)$ between the input $\omega(t)$ at time t and each of the weight vectors X_k is calculated. The winner weight $v(t)$ at time t is then defined as:

$$v(t) = \arg\min_{1 \le k \le N} d(\omega(t), X_k) \qquad [1]$$

For real-valued data, a possible distance measure $d(\cdot,\cdot)$ could be the Euclidean distance. When the winner has been found each weight vector is then updated as:

$$X_k(t+1) = X_k(t) + \alpha(t)h(k, v(t))(\omega(t) - X_k(t)) \qquad [2]$$

where $\alpha(t) \in [0,1]$ is a decreasing function of time which determines the learning rate of the SOM and h is a neighbourhood function which is a function of the distance on the lattice between the winner weight and the updated weight, that is: $h(k, v(t)) = h(d_L(i_{v(t)}, i_k))$.

Figure 2 shows a typical form 21 of the neighbourhood function h used in the SOM algorithm. The function h is positive, having a maximum value of 1 for $d_L = 0$ and decreasing monotonically towards 0 as d_L increases. In practical situations h is often Gaussian, i.e., of the form:

$$h(d_L) = e^{-ad_L^2} \qquad [3]$$

... for some appropriately chosen scaling factor a which may be a function of time. The SOM map is built iteratively. In other words, the steps in equations [1] and [2] are repeated for each t. For uniformly-distributed input data and a two-dimensional lattice with 15 × 15 weights, the SOM algorithm gives a result as in Figure 3.

Figure 3 shows a plot of the weight values on the support [0,1]×[0,1] of the input, where adjacent weights on the lattice are connected by a line segment in the plot. None of the line segments intersect, which indicates that the weights are organized. The weight values are spread uniformly over the input space and approximate the uniform distribution of the input data. To arrive at such a result the learning is divided into two phases. In the first phase, there is a large α and a, and the algorithm is in the self-organizing phase where the weights become organized. The second phase is the convergence phase where the neighbourhood width is reduced to zero as a increases and α approaches zero. The weight vectors converge to the approximation of the probability distribution of the input.

Unsupervised Clustering Based on the SOM

After the description of the basic SOM algorithm, some techniques of determining cluster centres will be disclosed. Consider the SOM shown in Figure 1. Figure 1 shows data points generated from six different normal distributions with different means and variances and plotted on this are the weight vectors from the SOM when data from this distribution was used as an input. It is seen that the weight vectors have organized and are more concentrated at the centres of the clusters. Hence the approximation of the probability distribution of the input. Reference numerals 13_1 to 13_6 denote six circles located at the cluster centres. The circles are not normally part of the

SOM. While it is easy for humans to determine where the cluster centres are, this task is surprisingly difficult for computers.

First, a probabilistic algorithm for determining the cluster centres will be briefly disclosed. Figure 4 shows a plot of the calculated probability of each weight vector being chosen as winner during the simulation. The probability for each weight vector is determined by keeping a record of the number of times the weight vector was chosen as winner and then dividing the number by the total number of iterations in the simulation. The i, j axis indicate the position of each weight on the SOM lattice and the $p(i, j)$ axis shows the probability of each of the weights. From this probability distribution the cluster structure is visible, where local maximum in the surface correspond to weight vectors positioned at input data cluster centres and the local minimum form a boundary between the local maximum and hence the clusters. However, the different clusters are not clearly distinct and the surface is not smooth. This roughness of the surface and variations in the peak values of the local maximum can lead to problems when trying to automatically determine the cluster centres using an algorithm which would associate a local maximum of the probability with a cluster centre. Defining a global threshold above which a weight vector would be considered a local maximum and a second global threshold below which a weight vector would be considered a local minimum leads to problems. For example, if in one part of the lattice the probability at a local maximum is below the global threshold to be considered a local maximum hence a cluster centre could lead to erroneous choice of cluster centres. In practise, as in the example shown, this is likely to occur. The alternative would be to define local thresholds for determining local maximum and minimum. However this would be a complicated process requiring an involved computation with no guarantee of a correct result .

A clustering algorithm according to a preferred embodiment of the invention is based on this observation. In effect the algorithm provides a means to smooth the probability surface just described. The use of a neighbourhood function means that the smoothing operation is done locally, emphasising local maximum as well as local minimum. The positive elements of the neighbourhood function emphasise the local maximum and the negative components emphasise the local minimum. The result is that the local maximums all reach consistently high values and local minimum reach consistently low values. This allows for the use of a global threshold to identify

the maximum and minimum and hence facilitating the use of a computer in the process. Hence instead of using directly the probability of a weight being the winner, a measure somehow related is used in the proposed algorithm which is described as follows.

For each weight i define a scalar coefficient C_i. This coefficient is bounded to the interval [0, 1] and its initial value before training can be quite small. The SOM algorithm is carried out as described earlier. At each iteration the winner weight $v(t)$ is determined as in equation [1] and each SOM weight i is updated according to equation [2]. At the same time each coefficient C_i is updated as:

$$C_i(t+1) = C_i(t) + C_{v(t)}(t)h_m(d_L)\delta \qquad [4]$$

where h_m is the second neighbourhood function. d_L and $v(t)$ are the same terms that were used in the SOM algorithm, that is, d_L is the distance on the lattice between node i and the winner node $v(t)$. δ is a small step value for adjusting convergence speed. δ is somewhat analogous to the α in the SOM algorithm.

Following the update then $C_i(t+1)$ is forced within the interval [0, 1]. For example, if $C_i(t+1) > 1$, it can be set to 1, and if $C_i(t+1) \leq 0$, it can be set to 0.01. The Hebbian-type learning is clear as the update of $C_i(t)$ depends on the value of $C_{v(t)}$.

Figure 5 shows a preferred form 51 of the second neighbourhood function h_m. Actually, Figure 5 shows a time-dependent version of the second neighbourhood function h_m such that the function is more pronounced as the number of prior iterations increases. In other words, with increasing time, the function h_m achieves negative values sooner (as the distance d_L increases), and the negative values are much more negative than during the earlier iterations.

As can be seen, the preferred form 51 of the second neighbourhood function h_m somewhat resembles the first neighbourhood function h used in the SOM algorithm. Like the first neighbourhood function h, the second neighbourhood function h_m starts at 1 when the distance is zero. Also, h and h_m both approach zero as the distance d_L increases.

But the second neighbourhood function h_m is preferably negative for some distances. For instance, in a 10 by 10 lattice, h_m may be negative for distances over 3. The negative value of the second neighbourhood function h_m

can be seen as a form of lateral inhibition between the weights. Lateral inhibition is a mathematical model that tries to approximate real biological phenomena. Similar to the h function used in the SOM, weights adjacent to the centre of activity have their coefficients and hence their activity increased, while the activity of weights further away from the centre of activity are inhibited.

However this lateral inhibition is rarely if ever used in practical applications. In the SOM the interaction between the weight vectors is defined by the neighbourhood function h defined by equation [3] which is strictly positive. If h was allowed to be negative at any point, divergence of the weight vectors could result, instead of convergence. In the clustering method proposed here this lateral inhibition is used to determine the cluster centres.

Intuitively it can be seen that if a weight i is quite often the winner then C_i will increase along with its neighbours. At the same time for i the winner, and for j far from i on the lattice C_j will be decreased. Similarly, if i is not often the winner its C_i will not increase very much and will be decreased by other winners at a distance on the lattice. Given the example in Figure 1 it is clear that weights in the inter-cluster regions will have a lower probability of being winners, whereas weights close to the cluster centres will have a higher probability of being winners. Hence it would be expected that the coefficients C_i would be larger for weights positioned in or near the cluster centres and small for positions between clusters. This would then provide boundaries between the cluster centres. Of course this depends on the fact that the weight vectors X_i reach an organized configuration.

Figure 6 shows an example of a computer pseudocode listing for generating the function shown in Figure 5.

Figure 7 shows a plot 70 of the coefficient values $C(i, j)$ for the weights i, j in the SOM of Figure 1 with $\delta = 0.01$ after 20000 iterations. The second neighbourhood function h_m varied with time and initially did not have a high level of lateral inhibition. Towards the end of the simulation, the level of lateral inhibition was increased. The surface shown in Figure 7 has six distinct and separated elevated regions. As a result of forcing the C_i between 0 and 1, each elevated region corresponds to a set of adjacent weights on the lattice whose coefficients C_i have saturated at or near 1 and whose weights X_i are found at the cluster centre. These regions are surrounded by regions where the coefficients C_i have been driven to small values. Hence it is possible to

determine which weights represent the same cluster. To determine which cluster an input vector belongs to, simply find the closest centre and assign the input to the cluster to which the centre belongs. Thus unlike the K-means algorithm where one weight represents the cluster, in the algorithm proposed here a group of weights represents the cluster. The means of classification is the same. Because the coefficients C_i are saturated near 0 or 1, it is a simple task to determine the cluster centres using a global threshold which could be set for example at a value of 0.5.

It should be noted that the plot 70 is for visualization purposes only and is not required by computers. Instead, reference number 71 points to an array of current coefficients $C_i(t)$ and reference number 72 points to an array of next coefficients $C_i(t+1)$. It is the array 72 of updated coefficients that a computer uses to determine the cluster centres and their locations. The arrow 806 denotes updating of the coefficients that takes place in step 806 of Figure 8 that will be described next.

Figure 8 is a flow chart illustrating a method according to a preferred embodiment of the invention wherein the method comprises two iterative processes 81 and 82 run in tandem. The odd-numbered steps on the left-hand side of Figure 8 relate to the known SOM algorithm. The even-numbered steps on the right-hand side of Figure 8 relate to the inventive algorithm for maintaining and updating the second data structure that is used to determine the cluster centres automatically. In step 801 the SOM algorithm is initialized. The initialization comprises selecting initial values for the α, a, h and randomly initializing the weight vectors X_i. Since Figure 8 shows an online algorithm, the values of the inputs $\omega(t)$ are not known at this stage. Step 802 is a corresponding initialization step for the second data structure. Steps 803, 805, 807 and 809 form the conventional SOM iteration. In step 803, input $\omega(t)$ is presented to the SOM at iteration t. In step 805, the winner weight $v(t)$ is selected according to equation [1]. In step 807, the weight vectors are updated according to equation [2]. In an optional step 807', the variables α that determines learning speed and/or the a of the first neighbourhood function h are updated. In step 809, the iterative loop is repeated until some predetermined stopping criteria have been met. For instance, the loop may be set to run a predetermined number of times, or the loop may be interrupted when each succeeding iteration fails to produce a change that exceeds a given threshold.

Steps 806 and 808 relate to the second iterative process 82 for maintaining and updating the second data structure that is used to determine the cluster centres. In step 806, the coefficients C_i are updated on the basis of the winner weights $v(t)$ according to equation [4]. In an optional step 808, parameters for the second neighbourhood function hm are updated (see Figure 5). According to a preferred feature of the invention, steps 806 and 808 of the second iterative process 82 are interleaved with the steps of the first iterative process 81. In this way, step 806 utilizes intermediate calculation results of step 805. Similarly, step 808, in which the second neighbourhood function h_m is updated, may utilize data from step 807 that updates the variable α and the first neighbourhood function.

Figures 9 and 10 show another pair of SOM map and coefficient map, respectively. In this example the number of clusters in the input data distribution was reduced by one and the same SOM algorithm was used. Figure 9 shows the result of both the input and the final configuration of the weight vectors. Figure 10 shows the resulting C_i values. The five cluster centres are once again quite clear. It is remarkable that the same algorithm, without any adjustments, was capable of finding the new number of clusters and the locations of those clusters.

The way the invention works is as follows. In the beginning there is a predefined lattice, which in this case is a two-dimensional. Each point of the lattice is given a label, e.g. (2,3), (0,15). This lattice remains fixed and the labelling of the lattice points does not change. In the above examples, the lattice structure is a 15×15 lattice. It is from the lattice that the distances dL used in all neighbourhood functions is determined, e.g. the distance between lattice points (1,3) and (7,8) could be 6, depending on which distance measure we use.

Each lattice point is associated with a weight vector. The dimension of the weight vector is always the same dimension as the input data vector. In the examples here the input data has two dimensions. It is the weight vectors that change depending on the input data point and the distance, on the lattice, between two lattice points, the first point being the lattice point associated with the winner and the second point, the lattice point associated with the weight vector to be updated. This distance is not used to update the weight vector directly, but is used to determine the value of the first neighbourhood function in the update of the weight vector.

Figures 1 and 7 show a two-dimensional plot of the input data points and the weight vectors from the SOM. The plot is in the input space and by chance all the input points were bounded by [-8, 8]. The data points are just the points shown. The weight vectors are plotted at the crossing point of the lines. Another way of looking at it is, a line is drawn between weight vectors whose associated lattice nodes are the closest nodes on the lattice. The fact that the plot appears as a lattice means that the weights are organized, that is, the weight vector associated with adjacent lattice nodes appear in the input data space as adjacent to each other.

The relationship between the coefficients and the SOM lattice is the same as the relationship between the weight vectors and the SOM lattice, except the dimension of the coefficients is always 1. It is somewhat similar, though not the same as a probability measure, where the probability would be that the weight vector associated to the lattice node to which the coefficient is associated will be chosen as the winner for any given data input. Another interpretation is that the coefficients somehow represent an exaggerated version of the probability distribution of the input data.

In conclusion, we might say that the lattice is a fixed structure. One weight vector is associated with each lattice node. The weight vector is in the input data space. Similarly, one coefficient is associated with each lattice node. It is a scalar value and represents an indication of probability, though not the real probability that the weight vector associated with the same lattice node will be chosen as winner for a given input data point.

The dimensionality of the input data and the lattice are not necessarily the same. The input data may have any number of dimensions, such as 5, 10 or 15. In that case the dimension of the weight vectors would also be 5, 10, or 15. But the lattice could still be two-dimensional. We could also choose a different lattice to begin with (i.e., change the SOM structure) and make it four-dimensional, for example. In this case, if we still choose to have 15 lattice nodes along each axis of the lattice then we would have 15×15×15×15 lattice nodes and associated with each lattice node a 5, 10 or 15-dimensional weight vector. The examples above use a two-dimensional lattice and a two-dimensional input space merely because it is easier to draw and visualize. In practical implementations one could expect that the input data has more dimensions but the lattice structure could be two-dimensional.. The

number of lattice nodes along each dimension of the lattice is variable depending on the amount of computational resources available.

Figure 11 shows a mixed clustering example consisting of two very different clusters, namely a first cluster 112 described by a normal distribution and a second cluster 114 described by a uniform distribution along a parabola. The same SOM was used as in the previous two examples. Figure 12 shows the map 120 of coefficients $C_{i,j}$ for this example. There are two distinct areas. There is a connected region 122 of high values around three edges of the lattice which correspond to the parabolic distribution cluster 112 in Figure 11, and the disc shape 124 of connected values which corresponds to the normal distribution 114. This example is quite interesting because of the complexity of the parabolic distribution. In the mixture model clustering algorithm, without any prior information it would be difficult to generalize this distribution to a normal distribution. Similarly, a K-means approximation would have difficulty resolving these two clusters as at some points the distance between two points in different clusters are smaller than between two points in the same cluster. In experiments on this example the best K-means came up with four clusters. This example indicates that the clustering algorithm according to the invention can be very general.

Automatic labelling of cluster centres

A further preferred embodiment of the invention relates to automatic and unsupervised labelling of the clusters. The same notation is used here as above and only notation pertinent to this embodiment will be explained. Consider a set of labels $B = \{1, 2, \ldots, K\}$ which will be used to label the clusters. In practise K should be at least greater than or equal to the expected number of clusters. In the case of no prior knowledge, it may be suitable to let $K = N$ the total number of weights in the SOM, as this imposes a limit on the maximum number of clusters which can be identified.

For each weight i in the SOM define a vector of coefficients Θ_i as:

$$\Theta i = (\theta_{i,1}, \theta_{i,2}, \ldots \theta_{i,K}) \qquad [5]$$

Each coefficient $\theta_{i,l} \in [0, 1]$ represents a weighting between the SOM node i and the label l. The weight i belongs to cluster l if:

$$l = \underset{1 \le k \le K}{arg\,max}\ \theta_{i,k} \qquad [6]$$

The updating algorithm used on these coefficients to achieve automatic labelling proceeds as follows. At time t SOM weight $v(t)$ is chosen as the winner. The weight and its neighbours are updated as in the normal SOM algorithm. Also the coefficient C_i is updated as well as the coefficients C_j of the neighbours of C_i. The updating of the coefficients C_i and the interpretation of the results form the basis of the main invention, namely the automatic and unsupervised clustering.

In the automatic and unsupervised labelling of the clusters, at the same time t the Θ_i are updated as follows. Define $lv(t)$ as the label of the cluster to which the winner weight $v(t)$ is assigned, thus from equation (6),

$$lv(t) = \underset{1 \le k \le K}{arg\,max}\ \theta_{v(t),k}(t) \qquad [7]$$

For all the weights j, $j = 1, \ldots, N$ then the components $\theta_{j,lv(t)}$ are updated as follows:

$$\theta_{j,lv(t)}(t+1) = \theta_{j,lv(t)}(t) + C_{v(t)}(t+1)\, h_B(d_L)\delta \qquad [8]$$

where once again h_B is a neighbourhood function and preferably has the form shown in Figure 13. The idea is that the neighbours of the winning weight will have their $\theta_{j,lv(t)}$ increased to ensure that they will be classified to the same cluster as the winner $v(t)$, and weights further away from the winner weight will have their $\theta_{j,lv(t)}$ decreased to ensure that they will be classified to a different cluster than the winner.

For the weights j where the neighbourhood function $h_B(d_L) > 0$ it is also advantageous to decrease the other coefficients $\theta_{j,lv(t)}$, $k = 1, \ldots K$, $k \neq lv(t)$ in Θ_j as follows:

$$\theta_{j,k}(t+1) = \theta_{j,k}(t) - C_{v(t)}(t+1) h_B(d_L)\delta \qquad [9]$$

This reinforces the labelling of the winner and its neighbours to the cluster label $lv(t)$.

Note that equation [4] uses C_v at iteration t whereas equations [8] and [9] use C_v at iteration t+1. Actually, the C_v coefficient changes so little between iterations that either value can be used, depending on which value is more conveniently available.

Figure 14 shows an example of the result of the combination of the SOM, automatic clustering and automatic labelling of the clusters for the case of an input distribution consisting of five normal distributions. In this case the set of labels was given by $L = \{1, 2, 3, 4, 5, 6, 7, 8\}$. The values of the $\theta_{i,j}$ were

randomly initialized. Figure 14 shows a SOM with five distinct regions, one region around each cluster. Each node in the five areas 1 to 5 is assigned a label such that nodes in area 1 have a label of 2, nodes in area 2 have a label of 4, etc. The inter-region areas have a label of 0. These weights had a maximum of Θ_i less than a threshold value of 0.2 and where not assigned as centres to any cluster. It is clear that the cluster centres have been properly labelled with the labels $\{2, 4, 5, 6, 8\}$.

Figure 15 shows how the automatic cluster-labelling algorithm can be integrated with the cluster-determination algorithm according to the invention. Figure 15 is a modification of Figure 8, with step 806 followed by step 806' that relates to the automatic cluster-labelling algorithm. In step 806, the cluster labels $lv(t)$ are determined according to equation [7]. Then the $\theta_{j,lv(t)}$ components are updated according to equation [8] and the $\theta_{j,k}$ components are updated according to equation [9]. By placing the step 806' inside the second iterative process 82, maximal synergy benefits are obtained, or in other words, the computational overhead is kept to a minimum because step 806' makes used of the winner selection and coefficient determination already performed for the SOM construction and the automatic cluster determination.

Summary

The technique according to the invention allows automatic determination of cluster centres with a minimal amount of information on the data. No explicit, initial estimate of the number of clusters is required. Given the nature of convergence of the SOM, there is no need to know the type of distribution of the clusters either. In this respect the algorithm is very general. However, although explicit initial estimates of the number of clusters are not required, care should be taken to ensure that the SOM lattice contains a number of nodes larger than the expected number of clusters, as well as choosing a non-monotonous neighbourhood function that is negative for large distances and provides a level of lateral inhibition to ensure the coefficients for the cluster regions stand out more clearly.

The preferred embodiment of the invention, in which the second iterative process is interleaved with the conventional iterative process, requires little computational overhead. Thus this embodiment invention is especially suitable for on-line application where human supervision is not available. Initial simulations on artificial data show that the inventive technique is simple and

apparently robust and is more easily generalized than most current clustering algorithms. The technique according to the invention can be considered somewhat as a hybrid of the K-means and probabilistic-model-based clustering.

It is readily apparent to a person skilled in the art that, as the technology advances, the inventive concept can be implemented in various ways. The invention and its embodiments are not limited to the examples described above but may vary within the scope of the claims.

CLAIMS

1. A computer-implemented method for determining cluster centres (13_1 - 13_6) in a first data structure (10), wherein the first data structure comprises a lattice structure (12) of weight vectors that create an approximate representation of a plurality of input data points (11);

the method comprising:

performing a first iterative process (81) for iteratively updating the weight vectors such that they move toward cluster centres (13_1 - 13_6);

performing a second iterative process (82) for iteratively updating a second data structure (70 - 72) utilizing results of the iterative updating of the first data structure; and

determining the weight vectors that correspond to cluster centres of the input data points on the basis of the second data structure (70 - 72).

2. A method according to claim 1, wherein each iteration in the first iterative process (81) comprises:

selecting a winner weight vector (v) for each data point on the basis of the distance between the data point and the weight vectors; and

calculating a next value of each weight vector on the basis of the current value of the weight vector and a first neighbourhood function (21, h) of the distance on the lattice structure between the weight vector and the winner weight vector; and

the second data structure (70 - 72) comprises a first coefficient (Ci) for each of the weight vectors in the lattice structure and each iteration in the second iterative process (82) comprises calculating (806) a next value of each first coefficient (Ci) on the basis of:

the current value of the first coefficient; and a combination of:

a first coefficient of the winner weight vector (v),

a second neighbourhood function (51, h_m) of the distance on the lattice structure between the weight vector and the winner weight vector, and

an adjustment factor (δ) for adjusting convergence speed between iterations.

3. A method according to claim 1 or 2, wherein the step of determining the weight vectors that correspond to cluster centres comprises selecting local maxima in the second data structure (70 - 72).

4. A method according to claim 2 or 3, wherein the combination is or comprises multiplication.

5. A method according to any one of claims 2 to 4, wherein the second neighbourhood function (51, h_m) is not monotonous.

6. A method according to any one of claims 2 to 5, wherein the first coefficients are limited to the range [0,1] and the second neighbourhood function (51, h_m) gives negative or positive values, respectively, for some distances.

7. A method according to any one of claims 2 to 6, wherein the second neighbourhood function (51, h_m) depends on the number of prior iterations.

8. A method according to any one of the preceding claims, wherein the input data points (11) represent real-world quantities.

9. A method according to any one of claims 2 to 8, wherein the first data structure (10) is or comprises a self-organizing map.

10. A method according to claim 9, further comprising:

estimating an upper limit K for the number of clusters in the self-organizing map;

defining a coefficient vector $\Theta i = (\theta_{i,1}, \theta_{i,2}, \ldots \theta_{i,K})$ for each weight vector i in the self-organizing map, the coefficient vector comprising K second coefficients $\theta_{i,l}$, each of which represents a weighting between the weight vector i and a label l; and

assigning cluster label l to weight vector i if:

$$l = \underset{1 \le k \le K}{arg\,max}\ \theta_{i,k}.$$

11. A method according to claim 10, wherein each iteration in the second iterative process (82) comprises calculating (806') a next value of each second coefficient on the basis of the current value of the second coefficient and a combination of:

a coefficient of the winner weight vector,

a third neighbourhood function (131, h_B) of distance, and

an adjustment factor (δ) for adjusting convergence speed between iterations.

12. A computer-readable program product comprising a computer program code, wherein executing the computer program code in a computer causes the computer to carry out the steps of the method according to claim 1.

(57) ABSTRACT

A computer-implemented method for determining cluster centres in a first data structure, wherein the first data structure comprises a lattice structure of weight vectors that create an approximate representation of a plurality of input data points. The method comprises performing a first iterative process (81) for iteratively updating the weight vectors such that they move toward cluster centres; performing a second iterative process (82) for iteratively updating a second data structure utilizing results of the iterative updating of the first data structure; and determining the weight vectors that correspond to cluster centres of the input data points on the basis of the second data structure.

(Figure 8)

Fig. 1

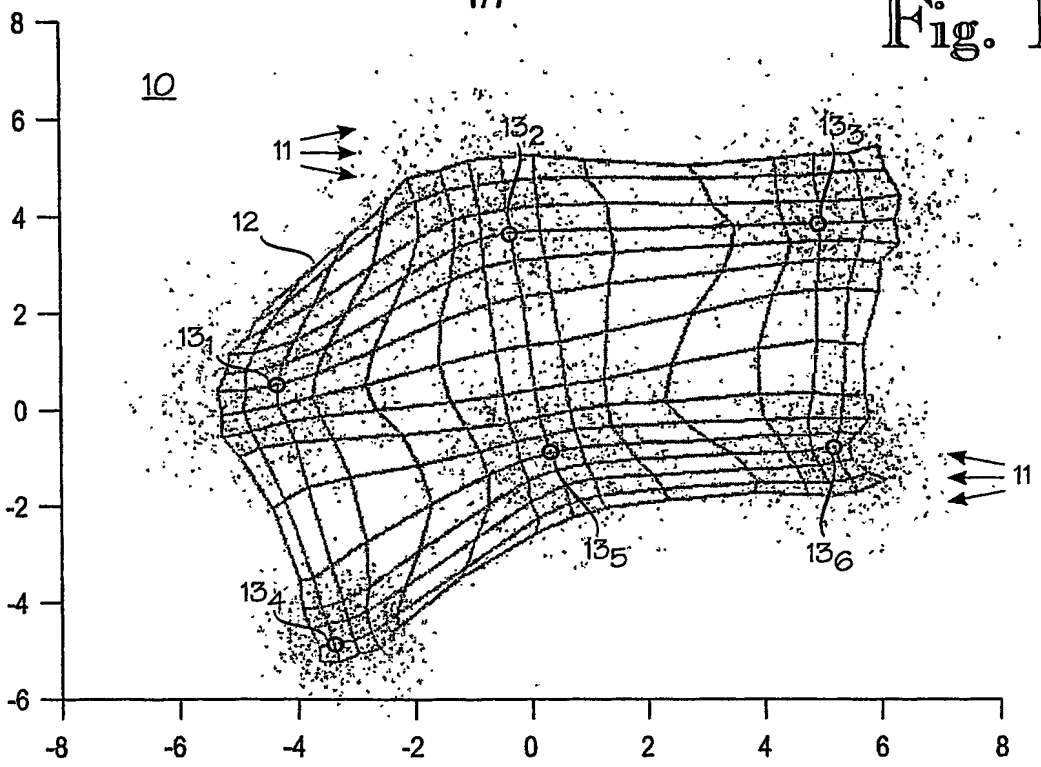


Fig. 2

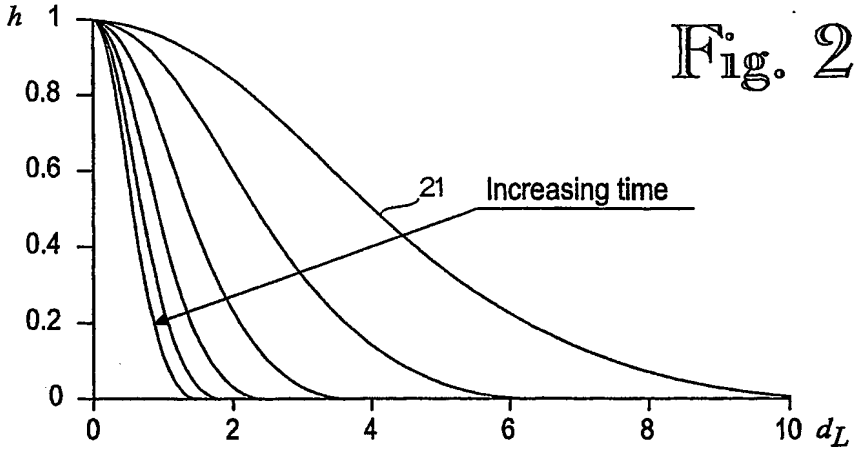


Fig. 3

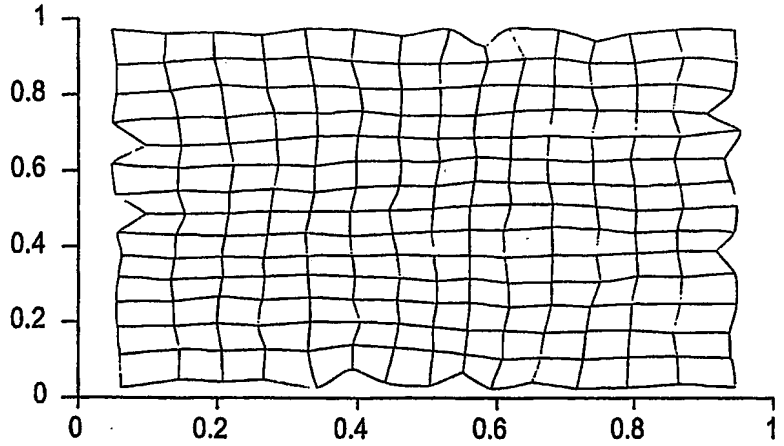


Fig. 6

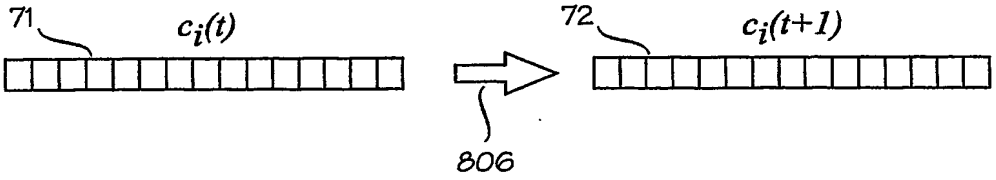
```
x=linspace(0,10,100);

figure

for i=1:1.5:10
    p=0.3+(0.4*(1-(i/10)));
     y=exp(-0.03*x.*x*i).*(p-(0.01*i*x.*x))/p;
     plot(x,y)
     hold on
end
```

60

Fig. 7



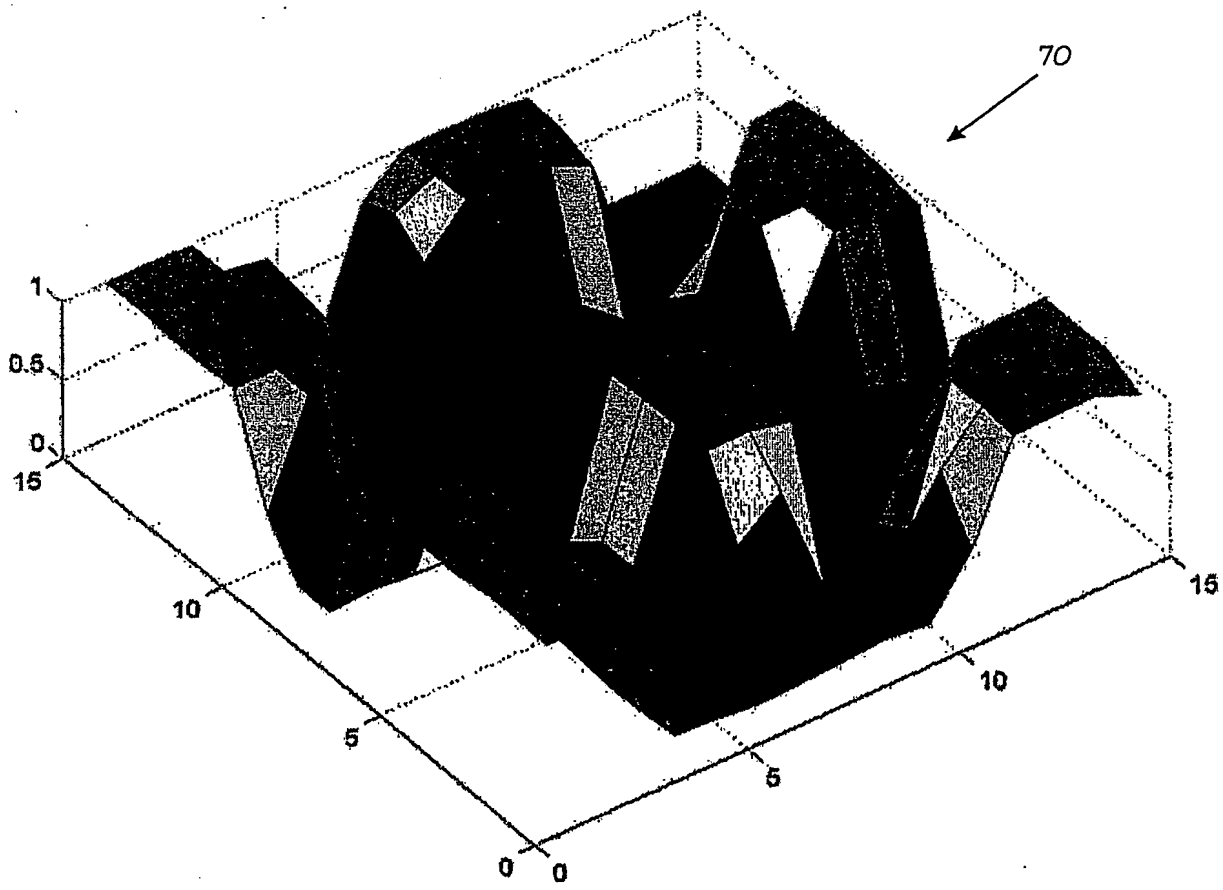


Fig. 8

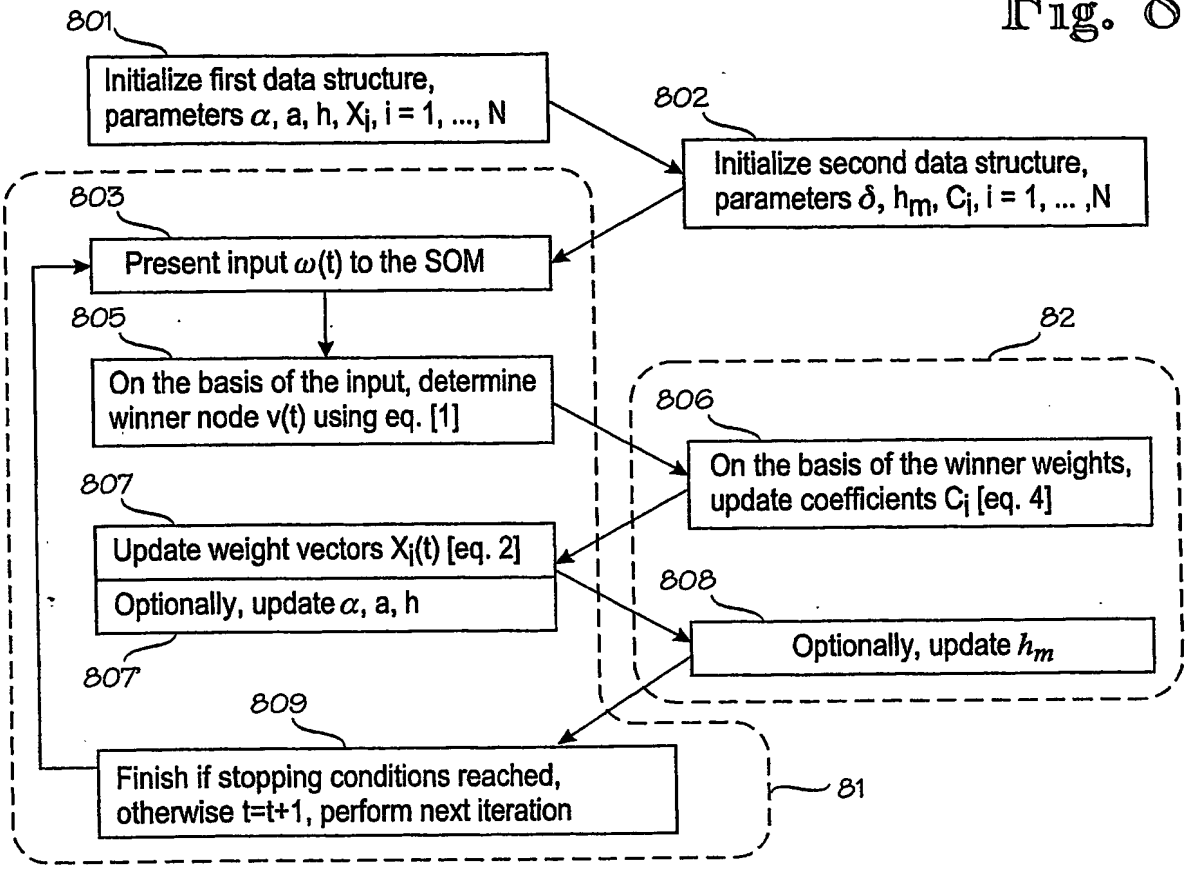
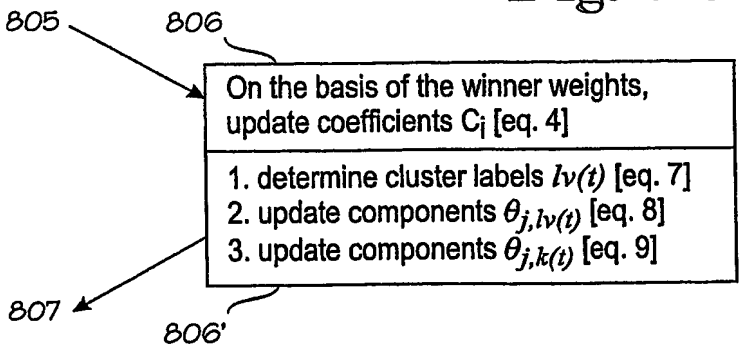

801
Initialize first data structure, parameters α, a, h, X_i, i = 1, ..., N
802
Initialize second data structure, parameters δ, h_m, C_i, i = 1, ... ,N
803
Present input ω(t) to the SOM
805
On the basis of the input, determine winner node v(t) using eq. [1]
82
806
On the basis of the winner weights, update coefficients C_i [eq. 4]
807
Update weight vectors X_i(t) [eq. 2]
Optionally, update α, a, h
807'
808
Optionally, update h_m
809
Finish if stopping conditions reached, otherwise t=t+1, perform next iteration
81

Fig. 15


805
806
On the basis of the winner weights, update coefficients C_i [eq. 4]
1. determine cluster labels $lv(t)$ [eq. 7]
2. update components $\theta_{j,lv(t)}$ [eq. 8]
3. update components $\theta_{j,k(t)}$ [eq. 9]
807
806'

5/7

Fig. 9

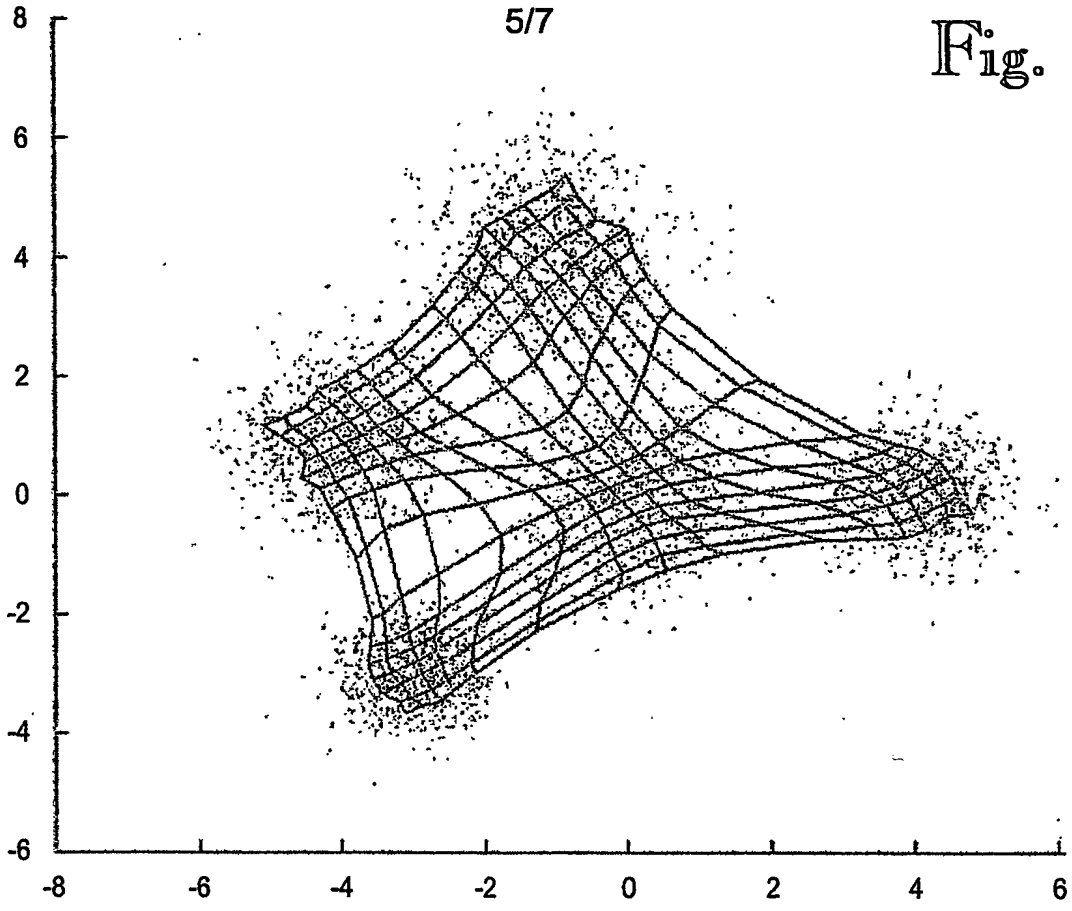


Fig. 10

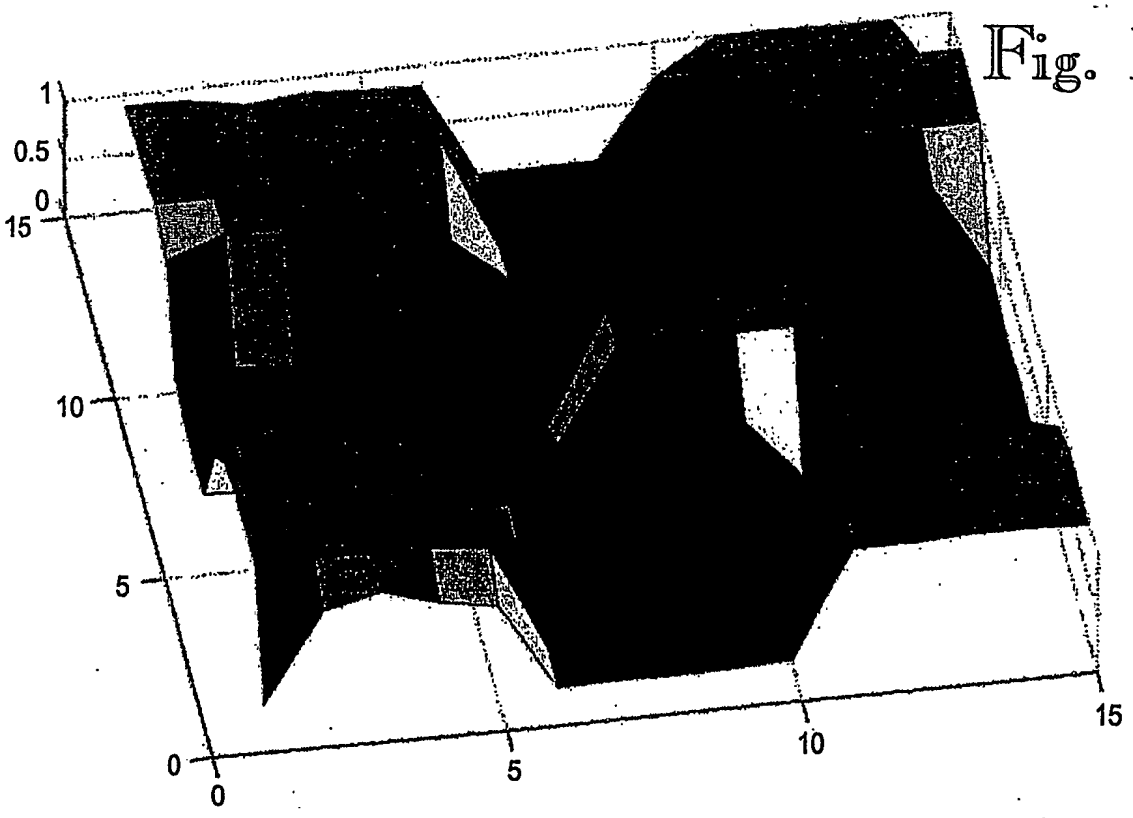


Fig. 11

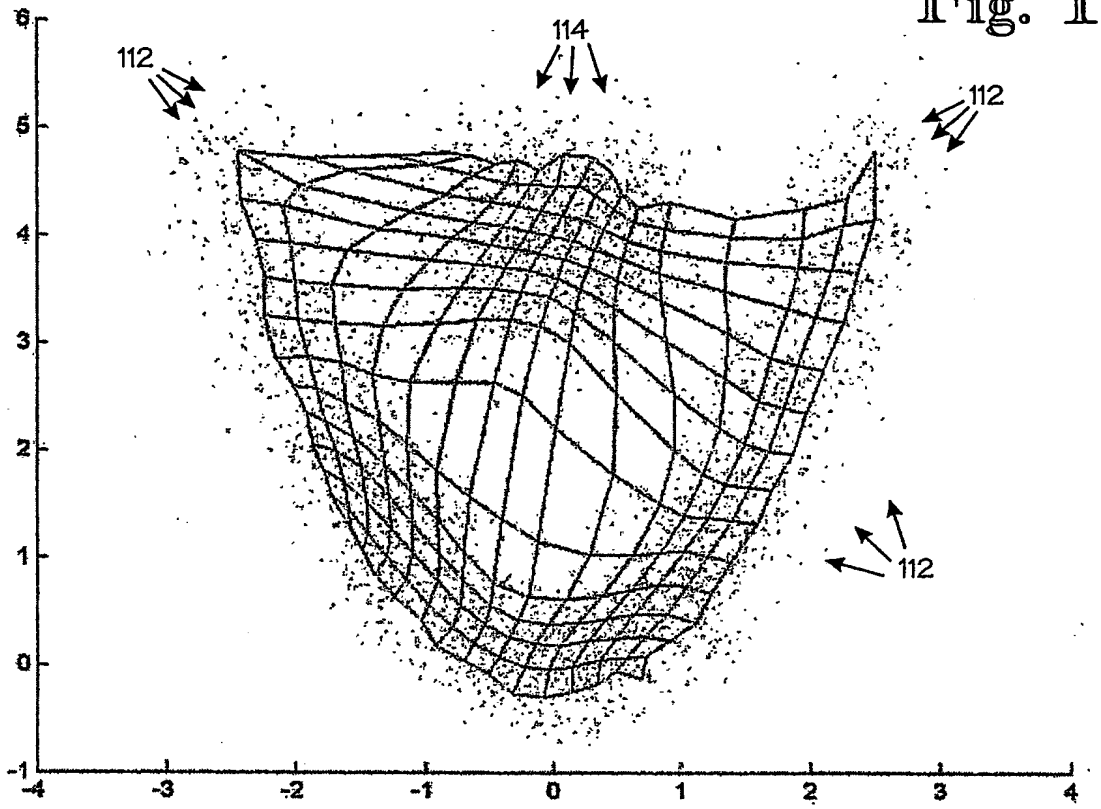

112
114
112
112
6
5
4
3
2
1
0
-1
-4
-3
-2
-1
0
1
2
3
4

Fig. 12

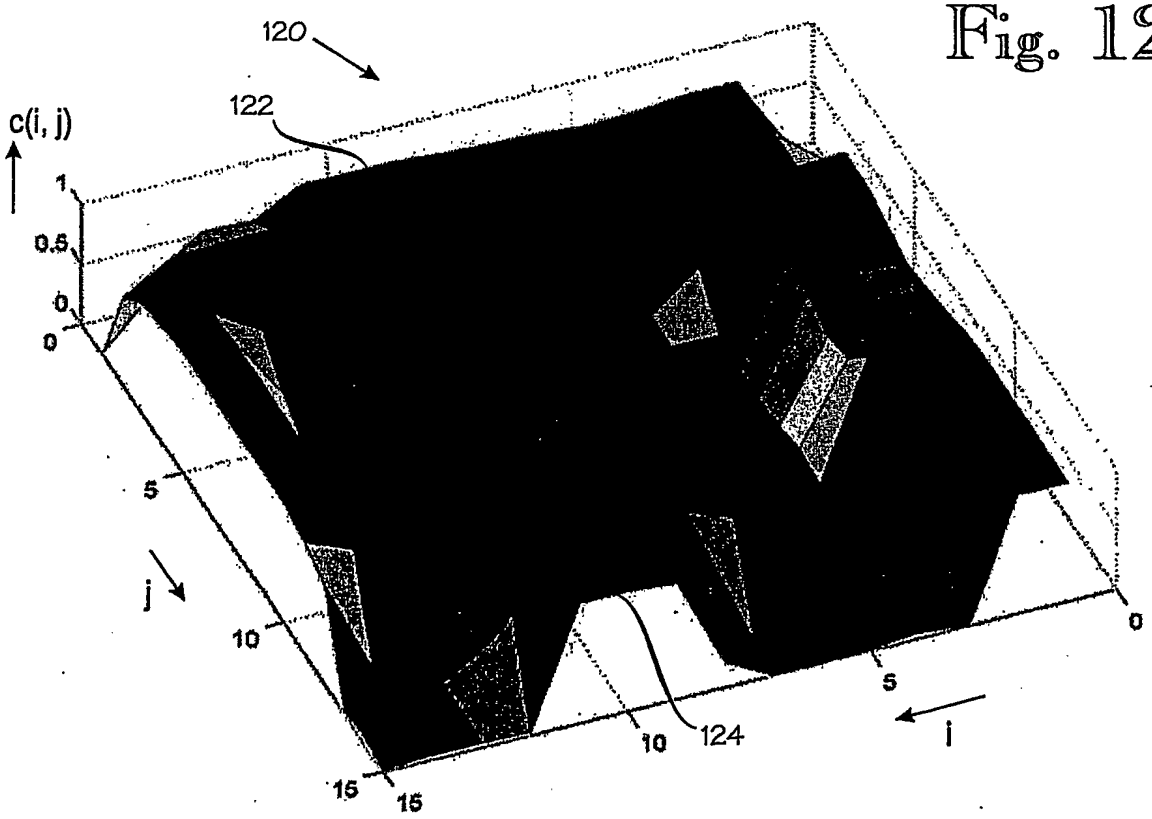

120
122
c(i, j)
1
0.5
0
0
5
j
10
15
15
10
124
5
i
0

Fig. 13

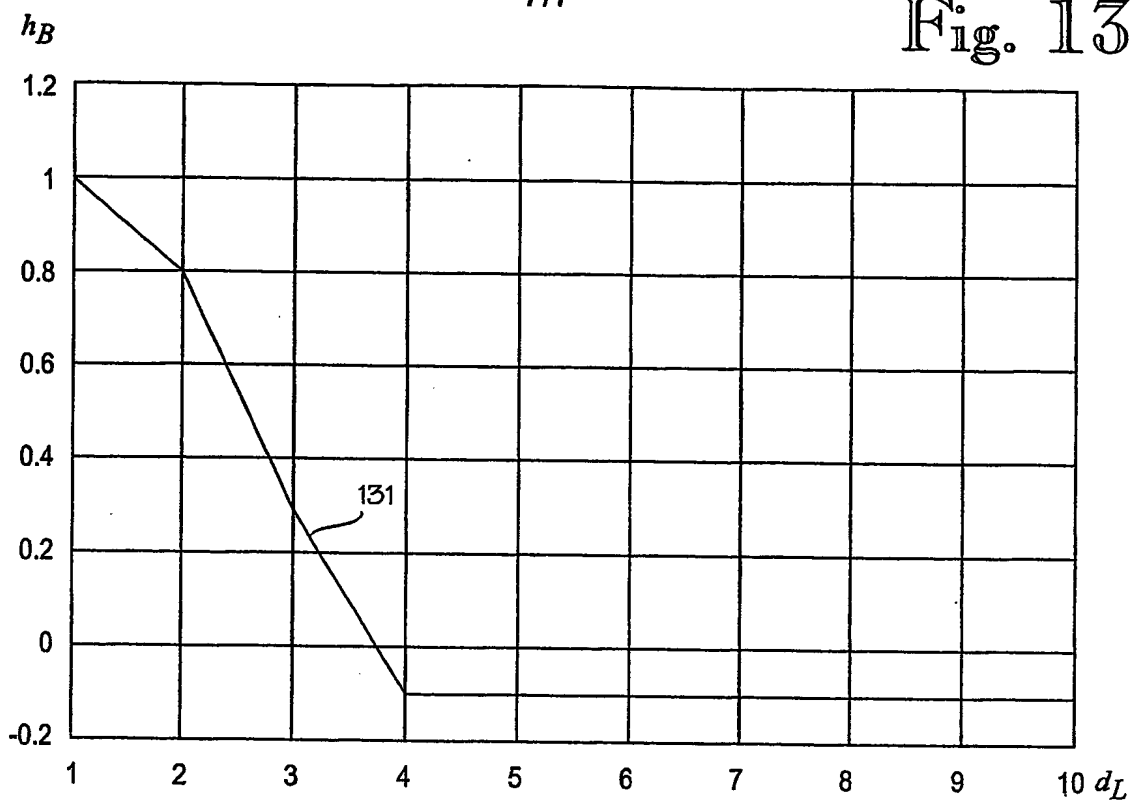
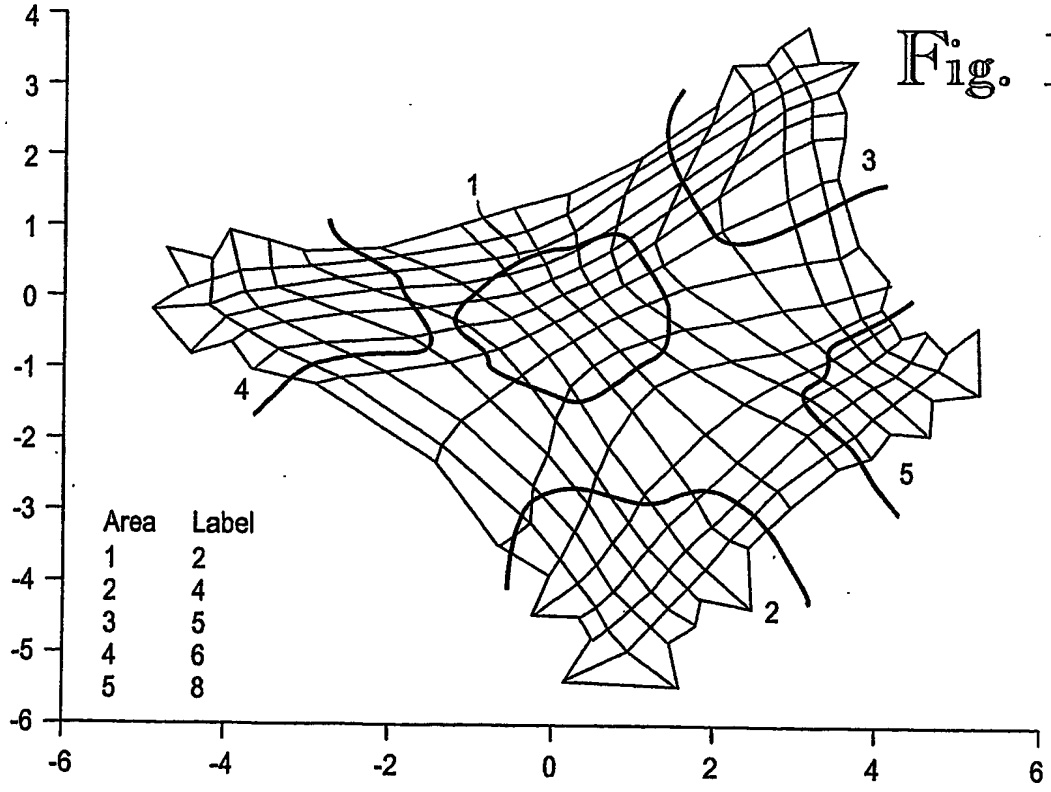

h_B
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
131
1
2
3
4
5
6
7
8
9
10 d_L

Fig. 14


4
3
2
1
0
-1
-2
-3
-4
-5
-6
1
3
4
5
2
Area Label
1 2
2 4
3 5
4 6
5 8
-6
-4
-2
0
2
4
6